



FIGURE 1

```

public class Valuation {
    private java.util.Calendar valuationDate;
    private double value;
    private MarketData

    public Valuation() {
    }

    public java.util.Calendar getValuationDate() {
        return valuationDate;
    }

    public void setValuationDate(java.util.Calendar valuationDate) {
        this.valuationDate = valuationDate;
    }

    public double getValue() {
        return value;
    }

    public void setValue(double value) {
        this.value = value;
    }
}

```

FIGURE 2

```
[Serializable]
public class Valuation {
    public DateTime valuationDate;
    public double value;
}
```

FIGURE 3

```

/**
 * JavaDealValuatorProxy.java
 * This file was autogenerated by the DataSynapse GridServer Manager.
 *
 * Be careful when making modifications to this class or associated generated classes. *
 * If you change the package, you must also change the packages
 * passed into the setXmlBeanDiscovery() method in the proxy class accordingly.
 */
package examples.dealvaluator.client.java;

import com.datasynapse.gridserver.client.*;

public class JavaDealValuatorProxy extends ServiceBindingStub {
    /**
     * Constructor. Any arguments to the initMethod, if set, are passed into this constructor.
     * @throws GridServerException if there is a problem creating the service, such as when there is no registered service
     with this name
     */
    public JavaDealValuatorProxy(examples.dealvaluator.client.java.MarketData in0) throws Exception {
        super("JavaDealValuatorExample", new Object[] { in0 }, null, null);
        String[] packages = new String[] { "examples.dealvaluator.client.java" };
        setXmlBeanDiscovery(packages,
            this.getClass().getClassLoader());
    }
    /**
     * Constructor. Any arguments to the initMethod, if set, are passed into this constructor.
     * @param options Additional service options. These options cannot override the options registered for this service
     * @param description Service description. These values cannot override the values registered for this service
     * @throws GridServerException if there is a problem creating the service, such as when there is no registered service
     with this name
     */
    public JavaDealValuatorProxy(examples.dealvaluator.client.java.MarketData in0, java.util.Properties options,
        java.util.Properties description) throws Exception {
        super("JavaDealValuatorExample", new Object[] { in0 }, options, description);
        String[] packages = new String[] { "examples.dealvaluator.client.java" };
        setXmlBeanDiscovery(packages, this.getClass().getClassLoader());
    }
}

```

FIGURE 4A

```

/**
 * Updates the state of the service instance by calling setMarketData
 * on any Engine prior to executing a request
 * @throws ServiceException if there is a problem with the service, such as it being cancelled or not started
 * @throws GridServerException on any other error
 */
public void setMarketData(examples.dealvaluator.client.java.MarketData in0) throws Exception {
    updateState("setMarketData", new Object[] {in0}, false);
}
/**
 * Synchronously calls valueOne
 * @return the result of the call
 * @throws ServiceInvocationException on an error executing the request
 * @throws ServiceException if there is a problem with the service, such as it being cancelled or not started
 * @throws GridServerException on any other error
 */
public examples.dealvaluator.client.java.Valuation valueOne(examples.dealvaluator.client.java.Deal in0,
java.util.Calendar in1) throws Exception {
    return (examples.dealvaluator.client.java.Valuation)execute("valueOne", new Object[] {in0, in1});
}
/**
 * Submits an asynchronous request to valueOne
 * @param callback If set, the callback is called upon completion of the request
 * @return The id of the request
 * @throws ServiceException if there is a problem with the service, such as it being cancelled or not started
 * @throws GridServerException on any other error
 */
public int valueOne(examples.dealvaluator.client.java.Deal in0, java.util.Calendar in1, Callback callback) throws
Exception {
    return submit("valueOne", new Object[] {in0, in1}, callback);
}
/**
 * Synchronously calls valueMany
 * @return the result of the call
 * @throws ServiceInvocationException on an error executing the request
 * @throws ServiceException if there is a problem with the service, such as it being cancelled or not started
 * @throws GridServerException on any other error
 */
public examples.dealvaluator.client.java.Valuation[] valueMany(examples.dealvaluator.client.java.Deal[] in0,
java.util.Calendar in1) throws Exception {
    return (examples.dealvaluator.client.java.Valuation[])execute("valueMany", new Object[] {in0, in1});
}

```

FIGURE 4B

```

/**
 * Submits an asynchronous request to valueMany
 * @param callback If set, the callback is called upon completion of the request
 * @return The id of the request
 * @throws ServiceException if there is a problem with the service, such as it being cancelled or not started
 * @throws GridServerException on any other error
 */
public int valueMany(examples.dealvaluator.client.java.Deal[] in0, java.util.Calendar in1, Callback callback) throws
Exception {
    return submit("valueMany", new Object[] {in0, in1}, callback);
}
/**
 * Convenience method for destroying this service instance.
 */
public void destroy() {
    super.destroy();
}
/**
 * Convenience method for waiting indefinitely until this service instance is inactive.
 * @throws InterruptedException if interrupted
 */
public void waitUntilInactive() throws InterruptedException {
    super.waitUntilInactive();
}
/**
 * Callback used for asynchronous execution
 */
public interface Callback extends ServiceBindingStub.AsyncCallback { public void handleResponse(
    Object response, int id);
    public void handleError(Exception e, int id);
}
}

```

FIGURE 4C

```

using System;

/// This file was autogenerated by the GridServer Manager.
///
/// Be careful when making modifications to this class or associated generated classes.
///
/// All of the following classes must remain within the same namespace.
///
namespace examples.dealvaluator.client.net {
    /// <summary>
    /// MarketData
    /// </summary>
    [Serializable]
    public class MarketData {
        public double rate;
    }
    /// <summary>
    /// Deal
    /// </summary>
    [Serializable]
    public class Deal {
        public string id;
        public int type;
        public double amount;
        public string data;
    }
    /// <summary>
    /// Valuation
    /// </summary>
    [Serializable]
    public class Valuation {
        public double val;
        public DateTime valuationDate;
    }
    /// <summary>
    /// NETDealValuatorProxy.cs
    /// </summary>
    public class NETDealValuatorProxy : DataSynapse.GridServer.Client.ServiceBindingStub {
        /// <summary>
        /// Constructor. Any arguments to the initMethod, if set, are passed into this constructor.
        /// </summary>
        public NETDealValuatorProxy(MarketData data) : base("NETDealValuatorExample", new object[] {data},
null, null, true) {}
        /// <summary>
        /// Constructor. Any arguments to the initMethod, if set, are passed into this constructor.
        /// </summary>
        /// <param name="options">Additional service options. These options cannot override the options registered
for this service</param>
        /// <param name="description">Service description. These values cannot override the values registered for
this service</param>
        /// <remarks>

```

FIGURE 5A

```

    /// Becuase options and descriptions are GridServer specific,
    /// this constructor is purposely commented out so that this class remains vendor-neutral
    /// </remarks>
    ///
    public NETDealValuatorProxy(MarketData data, DataSynapse.Util.Properties options,
DataSynapse.Util.Properties description) : base("NETDealValuatorExample", new object[] {data}, options, description, true) {}
    /// <summary>
    /// Updates the state of the service instance by calling setMarketData
    /// on any Engine prior to executing a request
    /// </summary>
    public void SetMarketData(MarketData data) {
        this.UpdateState("setMarketData", new object[] {data}, false);
    }
    /// <summary>
    /// Synchronously calls valueOne
    /// </summary>
    /// <returns>the result of the call</returns>
    public Valuation ValueOne(Deal deal, DateTime date) {
        object[] results = this.Invoke("valueOne", new object[] {deal, date});
        return (Valuation)results[0];
    }
    /// <summary>
    /// Begins an asynchronous request to valueOne
    /// </summary>
    /// <param name="callback">If set, the callback is called upon completion of the request</param>
    /// <param name="asyncState">The state object, which can be set to hold useful information which is passed
into the result</param>
    /// <returns>The result, which can be used to wait on the return.</returns>
    public System.IAsyncResult BeginValueOne(Deal deal, DateTime date, System.AsyncCallback callback,
System.Object asyncState) {
        return this.BeginInvoke("valueOne", new object[] {deal, date}, callback, asyncState);
    }
    /// <summary>
    /// Ends an asynchronous request to valueOne to extract the result
    /// </summary>
    /// <param name="asyncResult">The result, passed into the callback and returned from the Begin
call</param>
    /// <returns>the result of the request</returns>
    /// <remarks>
    /// If called from the callback, the result is immediately returned.
    /// If called immediately after the Begin call, this call blocks until completion.
    /// If the request resulted in an error, the exception is thrown from this method.
    /// </remarks>
    public Valuation EndValueOne(System.IAsyncResult asyncResult) {
        object[] results = this.EndInvoke(asyncResult);
        return (Valuation)results[0];
    }
}

```

FIGURE 5B


```

    /// <summary>
    /// Synchronously calls valueMany
    /// </summary>
    /// <returns>the result of the call</returns>
    public Valuation[] ValueMany(Deal[] deals, DateTime date) {
        object[] results = this.Invoke("valueMany", new object[] {deals, date});
        return (Valuation[])results[0];
    }
    /// <summary>
    /// Begins an asynchronous request to valueMany
    /// </summary>
    /// <param name="callback">If set, the callback is called upon completion of the request</param>
    /// <param name="asyncState">The state object, which can be set to hold useful information which is passed
into the result</param>
    /// <returns>The result, which can be used to wait on the return.</returns>
    public System.IAsyncResult BeginValueMany(Deal[] deals, DateTime date, System.AsyncCallback callback,
System.Object asyncState) {
        return this.BeginInvoke("valueMany", new object[] {deals, date}, callback, asyncState);
    }
    /// <summary>
    /// Ends an asynchronous request to valueMany to extract the result
    /// </summary>
    /// <param name="asyncResult">The result, passed into the callback and returned from the Begin
call</param>
    /// <returns>the result of the request</returns>
    /// <remarks>
    /// If called from the callback, the result is immediately returned.
    /// If called immediately after the Begin call, this call blocks until completion.
    /// If the request resulted in an error, the exception is thrown from this method.
    /// </remarks>
    public Valuation[] EndValueMany(System.IAsyncResult asyncResult) {
        object[] results = this.EndInvoke(asyncResult);
        return (Valuation[])results[0];
    }
    /// <summary>
    /// Convenience method for destroying this service instance.
    /// </summary>
    public void Destroy() {
        base.Destroy();
    }
    /// <summary>
    /// Convenience method for waiting indefinitely until this service instance is inactive.
    /// </summary>
    public void WaitUntilInactive() {
        base.WaitUntilInactive();
    }
}
}

```

FIGURE 5C

```
package example.services.adder.service;

public class JavaAdder {

    public double add(double a, double b) {
        return a + b;
    }
}
```

FIGURE 6

```
// Create a Service instance of JavaAdderExample
Service s = ServiceFactory.getInstance().createService("JavaAdderExample");

// Perform Synchronous add
Object[] arguments = new Object[] { new Double(5), new Double(6) };
Double sum = (Double)s.execute("add", arguments);
System.out.println("Result of add: " + sum);
```

FIGURE 7

```
// Handler for Service Clients
static class AdderHandler implements ServiceInvocationHandler {
    public void handleError(ServiceInvocationException e, int id) {
        System.out.println("Error from " + id + ": " + e);
    }
    public void handleResponse(Serializable response, int id) {
        System.out.println("Response from " + id + ": " + response);
        _total += ((Double)response).doubleValue();
    }
    public double getTotal() {
        return _total;
    }
    private double _total = 0;
}
}
```

FIGURE 8

```
// Perform Asynchronous adds
AdderHandler handler = new AdderHandler();
for (int i=0; i < 10; i++) {
    s.submit("add", new Object[] { new Double(i), new Double(i) }, handler);
}

// Wait until all the invocations have returned
s.waitUntilInactive(0);
System.out.println("Total: " + handler.getTotal());
```

FIGURE 9

```
// Create an instance of the Service Proxy
JavaAdderProxy adder = new JavaAdderProxy();

// Perform Synchronous add
double sum = adder.add(8.0, 10.0);
System.out.println("Result of Add: " + sum);
```

FIGURE 10

```
public double add(double in0, double in1) throws Exception {  
    return ((java.lang.Double) execute("add", new Object[] {new java.lang.Double(in0), new  
java.lang.Double(in1)})).doubleValue();  
}
```

FIGURE 11

```
public interface Callback extends ServiceBindingStub.AsyncCallback {  
    public void handleResponse(Object response, int id);  
    public void handleError(Exception e, int id);  
}
```

FIGURE 12


```
// Perform Asynchronous adds
AdderCallback callback = new AdderCallback();
for (int i=0; i < 10; i++) {
    adder.add(i, i, callback);
}

// Wait until all the invocations have returned
adder.waitUntilInactive();
System.out.println("Total: " + callback.getTotal());
```

FIGURE 13

```
JavaAdderProxy proxy = new JavaAdderServiceLocator().getJavaAdderProxy();
((org.apache.axis.client.Stub)proxy).setUsername(username);
((org.apache.axis.client.Stub)proxy).setPassword(password);
((org.apache.axis.client.Stub)proxy).setMaintainSession(true);

// Perform Synchronous add
double sum = adder.add(8.0, 10.0);
System.out.println("Result of Add: " + sum);
```

FIGURE 14

```

private void submitAll(JavaAdderProxy proxy) throws Exception {
    int num = 10;
    for (int i = 0; i < num; i++) {
        String id = proxy.add_Async(i, i);
        System.out.println("Submitted ID: " + id);
    }
}

private void collectAll(JavaAdderProxy proxy) throws Exception {
    Object result = null;
    int num = 0;
    while (true) { // providing a null id indicates that any available result should be collected
        String id = null;
        StringHolder idHolder = new StringHolder(id);
        try {
            result = proxy.collectAsync(idHolder);
        } catch (AxisFault ex1) {
            if (ex1.getFaultCode().getLocalPart().equals("Client")) {
                // Client types indicate an error executing the request on an Engine
                System.out.println("Received result #" + num++);
                handleError(ex1);
                continue;
            } else {
                // some other problem, such as unauthorized use, so just exit
                throw ex1;
            }
        }
        if (idHolder.value != null) { // if the returned id is set, then there is a result
            System.out.println("Received result #" + num++);
            handleResult(result, idHolder.value);
        } else { // either nothing currently available, or nothing outstanding
            if (result != null) {
                // if outstanding requests, result is time to wait before next collection
                Thread.currentThread().sleep( ( (Long) result).longValue());
            } else { // if both id and result are null, no outstanding requests.
                break;
            }
        }
    }
}
}

```

FIGURE 15

```
public class DBCalculator {  
    public void initDBConnection(Properties connProps, int someVal) {  
        _connection = new Connection(Properties props);  
    }  
  
    public Bar calculate(String arg1, ...) { ... }  
  
    public void close() {  
        if (_connection != null)  
            _connection.close();  
    }  
  
    private Connection _connection;  
}
```

FIGURE 16

```
public class DBCalculatorProxy {  
    public DBCalculatorProxy(Properties connProps, int someVal) { ... }  
    ...  
}
```

FIGURE 17

```

public class DBCalculator {
    ...
    public void stop() {
        _isStopped = true;
    }
    public Bar calculate(String arg1, ...) {
        while (_isStopped != null) {
            // iterate over looping variable
        }
    }
    private volatile boolean _isStopped = false;
}

```

FIGURE 18

```
public class BondValuationService {  
    public void addBonds(String[] bondIds) {  
        ...  
    }  
    public void setBonds (String[] bondIds) {  
        ...  
    }  
    public ValuationResults valueAllBonds(Scenario s) {  
        ...  
    }  
}
```

FIGURE 19

```

public class BondValuationServiceProxy {
    public void addBonds(String[] bondIds) {
        ...
    }
    public void setBonds (String[] bondIds) {
        ...
    }
    public ValuationResults valueAllBonds(Scenario s) {
        ...
    }
}

```

FIGURE 20


```
Service service = ServiceFactory.getInstance().createService("bondCalculator");

service.updateState("addBonds", new Object[] { bondIds }, false);
// or
service.updateState("setBonds", new Object[] { bondIds }, true);
```

FIGURE 21